```
Assignment -1


Apply·linear·regression·with·gradient·descent·to·predict·the·progression·of·
diabetes·in·patients.·(Dataset·Link·is·given·below).Learn·how·the·gradient·
descent·algorithm·works·and·finally·implement·it·on·any·new·data·set·and·make·
predictions.·Analyze·the·Learning·rate·and·step·size·values.
```

Assignment -1

Apply linear regression with gradient descent to predict the progression of diabetes in patients. (Dataset Link is given below).Learn how the gradient descent algorithm works and finally implement it on any new data set and make predictions. Analyze the Learning rate and step size values.

```
# Making the imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12.0, 9.0)
```

Double-click (or enter) to edit

The following data set provides the parameters of year of experience V/s Salary

Refrence 1: How to load data/fetch data set from url: https://towardsdatascience.com/4-awesome-ways-of-loading-ml-data-in-google-colab-9a5264c61966

Data set is fetch from the url: https://github.com/contactsunny/data-science-examples/blob/master/salaryData.csv

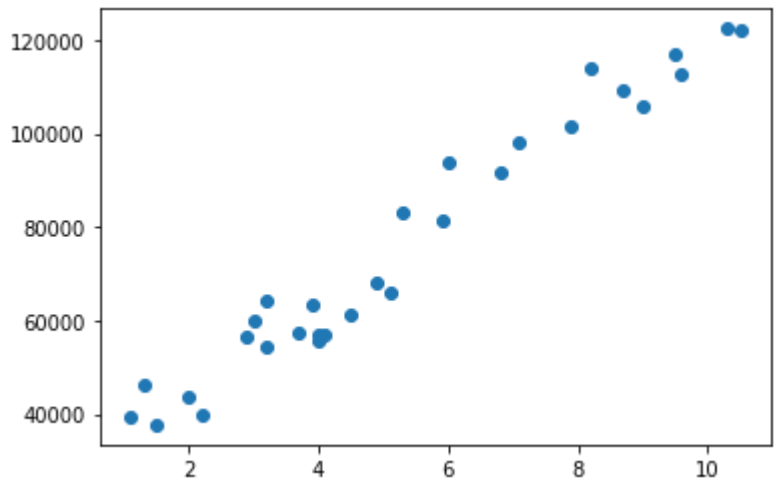
```
url = "https://raw.githubusercontent.com/contactsunny/data-science-examples/master/salaryData
data = pd.read_csv(url)
```

```
print(data)
print(data.size)
print(data.shape)
```

```
    YearsExperience    Salary
```

```
0              1.1   39343.0
1              1.3   46205.0
2              1.5   37731.0
3              2.0   43525.0
4              2.2   39891.0
5              2.9   56642.0
6              3.0   60150.0
7              3.2   54445.0
8              3.2   64445.0
9              3.7   57189.0
10             3.9   63218.0
11             4.0   55794.0
12             4.0   56957.0
13             4.1   57081.0
14             4.5   61111.0
15             4.9   67938.0
16             5.1   66029.0
17             5.3   83088.0
18             5.9   81363.0
19             6.0   93940.0
20             6.8   91738.0
21             7.1   98273.0
22             7.9  101302.0
23             8.2  113812.0
24             8.7  109431.0
25             9.0  105582.0
26             9.5  116969.0
27             9.6  112635.0
28            10.3  122391.0
29            10.5  121872.0
60
(30, 2)
```

```
X = data.iloc[:, 0]
Y = data.iloc[:, 1]
plt.scatter(X, Y)
plt.show()
```



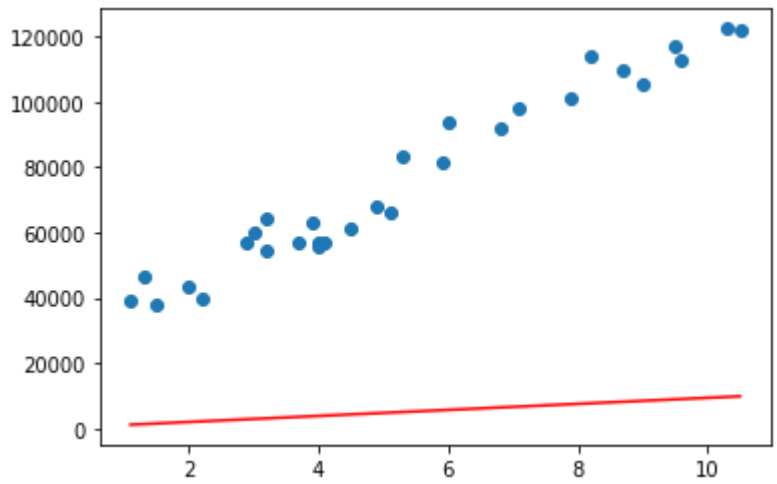
```
# Building the model
m = 0
c = 0

L = 0.0001  # The learning Rate
epochs = 10  # The number of iterations to perform gradient descent

n = float(len(X)) # Number of elements in X

# Performing Gradient Descent
for i in range(epochs):
    Y_pred = m*X + c  # The current predicted value of Y
    D_m = (-2/n) * sum(X * (Y - Y_pred))  # Derivative wrt m
    D_c = (-2/n) * sum(Y - Y_pred)  # Derivative wrt c
    m = m - L * D_m  # Update m
    c = c - L * D_c  # Update c

print (m, c)
```

```
923.7346095241653 147.39444789947996
```

y= mx+c

*CASE1: Learning rate= 0.001 and step size= 10 *

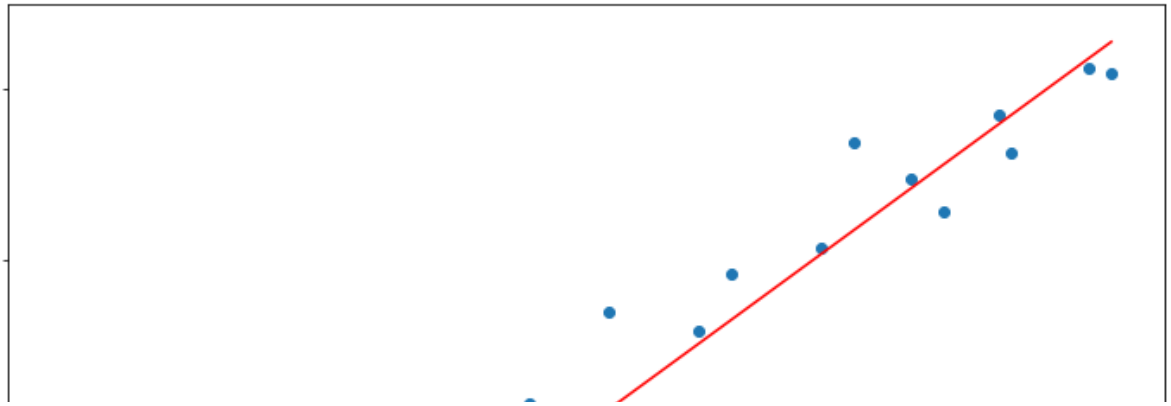
```
# Making predictions
Y_pred = m*X + c

plt.scatter(X, Y)
plt.plot([min(X), max(X)], [min(Y_pred), max(Y_pred)], color='red') # predicted
plt.show()
```



CASE 2 Learning rate= 0.015 and step size= 500

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12.0, 9.0)
url = "https://raw.githubusercontent.com/contactsunny/data-science-examples/master/salaryData
data = pd.read_csv(url)
X = data.iloc[:, 0]
Y = data.iloc[:, 1]
# Building the model
m = 0
c = 0

L = 0.015  # The learning Rate
epochs = 500  # The number of iterations to perform gradient descent

n = float(len(X)) # Number of elements in X

# Performing Gradient Descent
for i in range(epochs):
    Y_pred = m*X + c  # The current predicted value of Y
    D_m = (-2/n) * sum(X * (Y - Y_pred))  # Derivative wrt m
    D_c = (-2/n) * sum(Y - Y_pred)  # Derivative wrt c
    m = m - L * D_m  # Update m
    c = c - L * D_c  # Update c

print (m, c)
Y_pred = m*X + c

plt.scatter(X, Y)
plt.plot([min(X), max(X)], [min(Y_pred), max(Y_pred)], color='red') # predicted
plt.show()
```

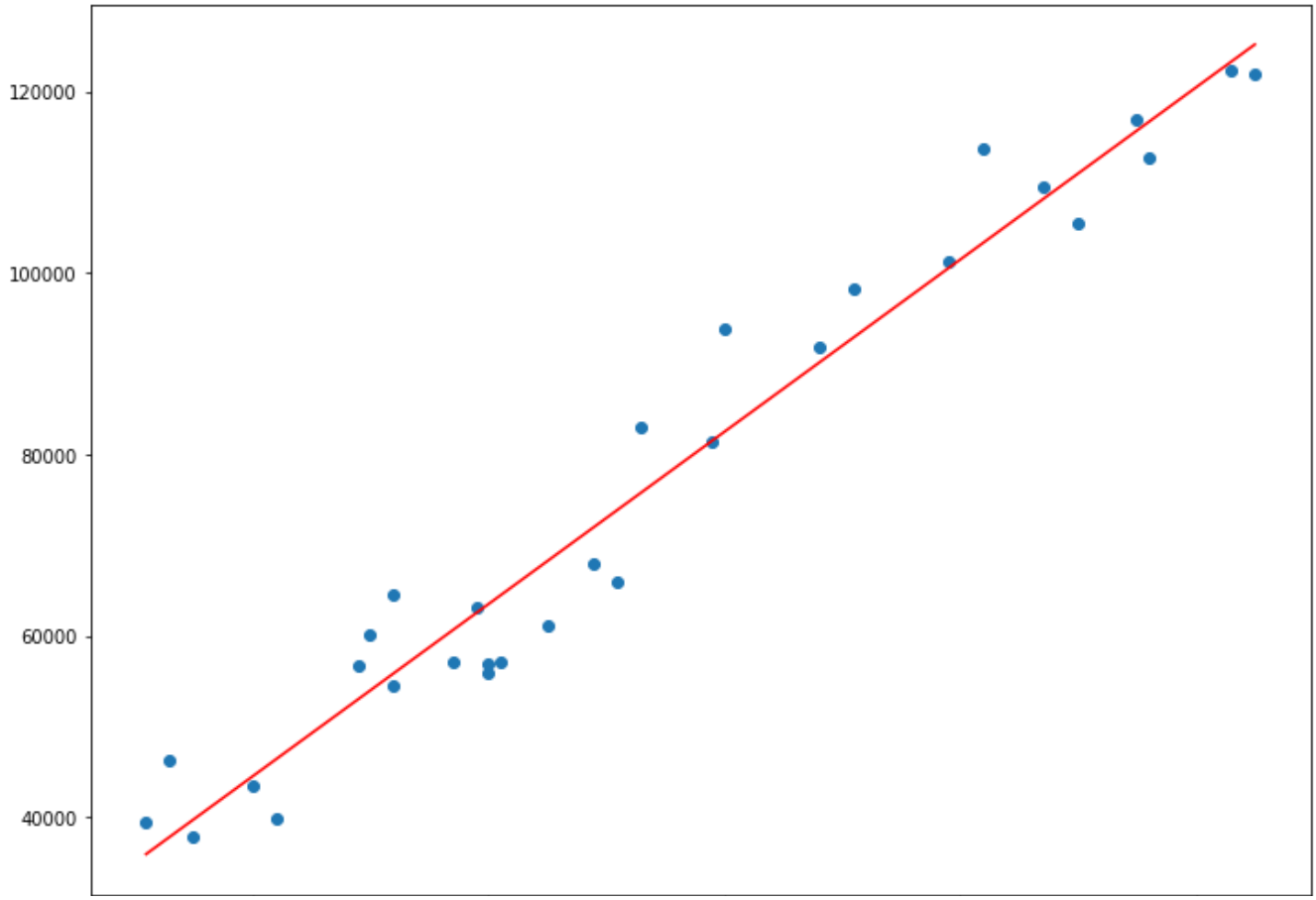
```
9596.796659300517 24802.72256813863
```



CASE 3 Learning rate= 0.01 and step size= 1000

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12.0, 9.0)
url = "https://raw.githubusercontent.com/contactsunny/data-science-examples/master/salaryData
data = pd.read_csv(url)
X = data.iloc[:, 0]
Y = data.iloc[:, 1]
# Building the model
m = 0
c = 0

L = 0.01  # The learning Rate
epochs = 1000  # The number of iterations to perform gradient descent

n = float(len(X)) # Number of elements in X

# Performing Gradient Descent
for i in range(epochs):
    Y_pred = m*X + c  # The current predicted value of Y
    D_m = (-2/n) * sum(X * (Y - Y_pred))  # Derivative wrt m
    D_c = (-2/n) * sum(Y - Y_pred)  # Derivative wrt c
    m = m - L * D_m  # Update m
    c = c - L * D_c  # Update c

print (m, c)
Y_pred = m*X + c

plt.scatter(X, Y)
plt.plot([min(X), max(X)], [min(Y_pred), max(Y_pred)], color='red') # predicted
plt.show()
```

```
9501.013219578916 25448.181745946968
```



Conclusion